

Modernizing Fortran 77 Legacy Codes

Viktor K. Decyk^{1,2} and Charles D. Norton²

¹*University of California at Los Angeles
Department of Physics and Astronomy
Los Angeles, CA 90095-1547*

²*National Aeronautics and Space Administration
Jet Propulsion Laboratory, California Institute of Technology
MS 168-522, 4800 Oak Grove Drive, Pasadena, CA 91109-8099*

Phone: 310-206-0371, Fax: 310-825-4057, E-mail: decyk@physics.ucla.edu

Over the course of 30 years, the scientific community has developed a large legacy of scientific programs, written in early versions of Fortran, which have a great deal of intellectual and commercial value. These codes have been carefully validated and often give excellent performance, even on modern computers. The early versions of Fortran (primarily Fortran 77) are simple languages useful at a time when the complexity of computing projects was limited. In fact, Fortran codes continue to be written and used, because compilers are very mature and highly optimized for numerical calculations. Furthermore, many scientific computational projects are relatively small (often less than 10,000 lines, rarely over 100,000 lines), and generally there is only one author involved.

Nevertheless, as computational power has increased, the ambitions of computational scientists have also increased, and there is a desire to model systems too complex to be written by a single author. For such projects, the early versions of Fortran are inadequate. Modern programming languages have evolved which have much better support for complex programming projects, among them object-oriented languages and Fortran 90. Should one abandon these legacy codes and rewrite everything in C++, or can we somehow continue to use the legacy inside a more modern structure?

Our approach is to build a modern superstructure around the older legacy code rather than a complete rewrite. We have developed a step-by-step process that allows software to be modernized, while also improving its quality. The application remains in productive use during the entire process. As much as possible, we do not modify the original subroutines, but rather incorporate the modern features in interface (wrapper) libraries. Our methodology is based on Fortran 90/95, because it has the modern features we desire, while still maintaining backward compatibility. We make use of Fortran 90 language features such as modules, derived types, and dynamic array objects. Embedding the older code inside the interface libraries encapsulates the implementation details of the legacy code, while adding dynamic features and additional safety checks. This also enables multiple authors to work on pieces of the code without interfering with one another and better reflects the problem domain. The new code can evolve toward an object-oriented design, if that is desired. Once the new superstructure works correctly, there is always the option of replacing individual pieces of the legacy code.

In our presentation, we will describe our methodology and demonstrate its success for plasma particle-in-cell codes for Tokamak modeling as well as for an optical modeling code important to NASA's Next Generation Space Telescope Project and Space Interferometry Missions.

References

1. Decyk, V. K., Norton, C. D., and Szymanski, B. K. How to support inheritance and run-time polymorphism in Fortran 90. *Computer Physics Communications*, 115:9-17, December 1998.
2. Decyk, V. K., Norton, C. D., and Szymanski, B. K. How to Express C++ Concepts in Fortran 90. *Scientific Programming*, 6(4):363-390, Winter 1997. IOS Press.

Acknowledgment: Work supported by NASA, NSF, and DOE.